**SIRALAMA VE FİLTRELEME (SORTING AND FILTERING)**

Şimdiye kadar, verilerimizi SELECT ve WHERE komutlarıyla nasıl tablolar halinde çekeceğimizi görmüştük. Çoğu zaman verilerimizi belli bir sıra ve düzen içerisinde listelemek isteriz. Bu metinsel ve sayısal değerlere dayalı artan ve azalan bir sıralama olabilir. Bu gibi durumlarda ORDER BY anahtar kelimesini kullanırız.

ORDER BY deyimi için sözdizimi aşağıdaki gibidir:

|  |  |
| --- | --- |
| 1  2  3  4 | **SELECT “kolon\_adlari”**  **FROM “tablo\_adi”**  **[WHERE “koşul”]**  **ORDER BY "sıralanacak\_kolon\_adi1" [ASC veya DESC]** |

Köşeli parantez [ ] işaretinin yukarıdaki sözdizimindeki anlamı içerisindeki koşulların isteğe bağlı olduğunu belirtir. Eğer WHERE koşulu var ise, ORDER BY ‘dan önce yazılmalıdır.

Verilerimizi sorgularımız ile listeledikten sonra bu verileri istediğimiz kolonlara göre sıralama yapabiliriz. Bunun için ORDER BY deyimini kullanırız. Eğer ORDER BY deyimini yalnız olarak kullanırsak verilerimiz varsayılan olarak artan şeklinde sıralanır. Yani alfabetik verilerde A’dan Z’ye, numerik verilerde ise küçükten büyüğe olacak şekilde sıralama gerçekleşir. Ama biz tam tersi yönde yani azalan şekilde sıralamak istiyorsak, alfabetik olarak Z’den A’ya, numerik olarak büyükten küçüğe yani bunun için DESCENDING deyimini veya DESC deyimini kullanırız. İkisi de aynı işlemi yapar. Artanın ayrıca deyimsel ifadesi ASCENDING veya ASC’dir. Kullanım kalıbı aşağıdaki gibidir:

**SELECT kolon\_adi1, kolon\_adi2**

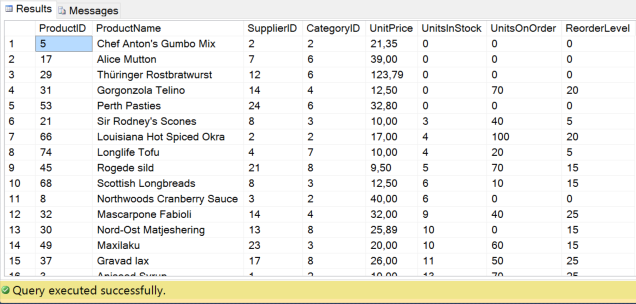
**FROM tablo\_adi**

**ORDER BY "kolon\_adi1" [ASC veya DESC]**

Örnekler ile konumuzu pekiştirelim:

|  |  |
| --- | --- |
| 1  2  3 | SELECT ProductID ,ProductName ,SupplierID ,CategoryID ,UnitPrice, UnitsInStock ,UnitsOnOrder ,ReorderLevel  FROM NORTHWND.dbo.Products ORDER BY UnitsInStock |

NORTHWND.dbo.Products tablosundan ProductID, ProductName, SupplierID, CategoryID, UnitPrice, UnitInStock, UnitsOnOrder, ReorderLevel kolonlarını, UnitsInStock kolonunun artan haline göre sıraladım.



Birden fazla kolona göre sıralama yapmak da mümkündür. Bu durumda ORDER BY deyimi şu şekilde kullanılır:

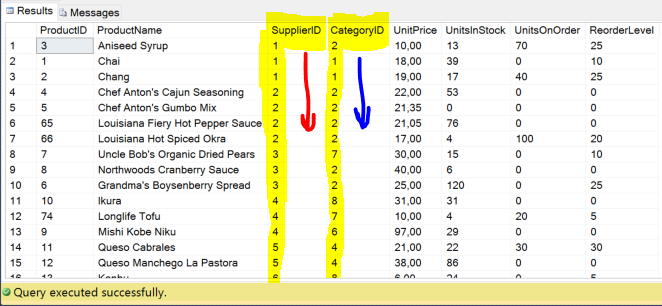
**ORDER BY "kolon\_adi1" [ASC, DESC], " kolon\_adi2" [ASC, DESC]**

Örnek:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT ProductID ,ProductName ,SupplierID ,CategoryID ,  UnitPrice, UnitsInStock ,UnitsOnOrder ,ReorderLevel  FROM NORTHWND.dbo.Products  ORDER BY SupplierID, CategoryID DESC |

Bu sorgumda da NORTHWND.dbo.Products tablosundan ProductID, ProductName, SupplierID, CategoryID, UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel alanlarını listele ancak listelerken sıralama SupplierID artan iken, CategoryID azalan olsun demektir. Peki biri artarken biri ne demek azalan olsun. Diyelim ki SupplierID alanında 1 ID’sine sahip 5 veri var. 2 ID’sine sahip 3 veri var ve böyle gidiyor. SupplierID alanında 1 ID’sine sahip 5 verinin kendi içinde sıralaması ise CategoryID azalan olacak şekilde yap demek oluyor. SQL’de query okuma işlemi soldan sağa doğru sıra ile giderdi, hatırlayınız. Tabi parantezler olursa durum değişir.

Görelim sonucu:



Görüldüğü üzere SupplierID alanı artan şeklinde, CategoryID azalan şekilde sıralanmış.

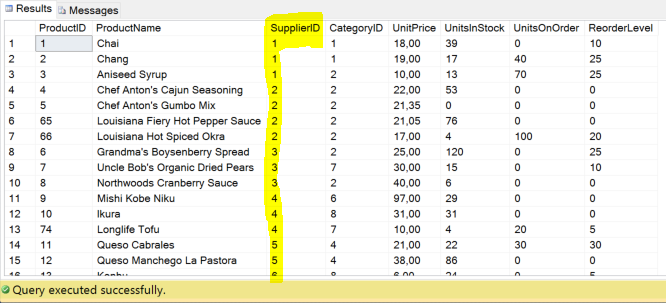
Sütun adının yanı sıra ORDER BY’ı uygulamak istediğiniz sütunun konumunu (SQL sorgusunu temel alarak) belirterek te kullanabilirsiniz. Sütun sıralaması ilk sütun 1. ikinci sütun 2. olarak gitmektedir. Yani aşağıdaki iki söz dizimi de aynı tabloyu verecektir.

Sutun Adı İle

|  |  |
| --- | --- |
| 1  2  3 | SELECT ProductID ,ProductName ,SupplierID ,CategoryID ,  UnitPrice, UnitsInStock ,UnitsOnOrder ,ReorderLevel  FROM NORTHWND.dbo.Products  ORDER BY SupplierID |

Sütun Sırası İle

|  |  |
| --- | --- |
| 1  2  3 | SELECT ProductID ,ProductName ,SupplierID ,CategoryID ,  UnitPrice, UnitsInStock ,UnitsOnOrder ,ReorderLevel  FROM NORTHWND.dbo.Products  ORDER BY 3 |



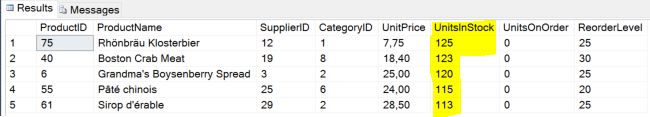
TOP SORGUSU

Veri tablomuzdan ya da veri tablolarımızdan belirli sayıda veri çekmek için top komutunu kullanırız.

Örnek:

Mesela stoklarımızda en fazla bulunun ilk 5 ürünü getirmesini isteyebilirim. Ve bu ürüneri elden çıkartmak için 5 ürünün ne olduğunu TOP sorgusuyla bulabilirim.

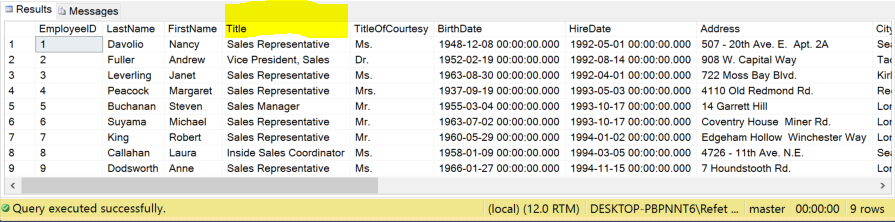
|  |  |
| --- | --- |
| 1  2  3  4 | SELECT TOP 5 ProductID ,ProductName ,SupplierID ,CategoryID ,  UnitPrice, UnitsInStock ,UnitsOnOrder ,ReorderLevel  FROM NORTHWND.dbo.Products  ORDER BY UnitsInStock DESC |



DISTINCT SORGUSU

Tablolarımızda tekrarlanan verilerimiz varsa ve bu tekrarlanan verileri istemiyorsak DISTINCT sorgusunu kullanırız.

Örnek:



Employees Tablosu

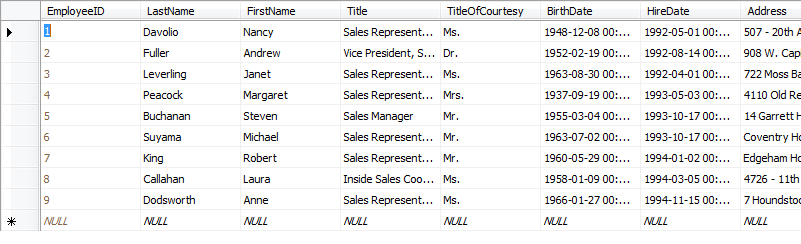
Şirketimizde kaç farklı pozisyonda çalışan işçimiz olduğunu öğrenmek için çalışanlar tablomuzdan unvanlarını listelemek isteyebiliriz.

|  |  |
| --- | --- |
| 1  2 | SELECT DISTINCT Title  FROM NORTHWND.dbo.Employees |

**OPERATORLER (OPERATORS)**

SQL’de de bir takım ifadeler ve operatörlere ihtiyaç duyulur. Yazılımda olduğu gibi matematiksel işaretlerin bir kısmı, bir takım mantıksal ifadeleri SQL ortamında da kullanmaktayız. Operatörler veri üzerinde işlem yaparak bir sonuç döndüren işlemlerdir. Operatörler özel karakterlerle veya özel anahtar kelimelerle temsil edilir.

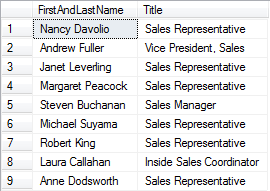
**+ Operatörü:** İki sayıyı ya da iki adet string ifadeyi toplamaya yarar. Örneğin + operatörünü Northwind veri tabanında Employees tablosunda kullanalım. Employees tablosu normalde şu şekilde görünüyor;

[](http://www.ismailgursoy.com.tr/wp-content/uploads/2012/02/29.024.png)

Görüldüğü gibi FirstName ve LastName ayrı kolonlarda. Şimdi ben bir sorgu yazacağım ve bu sorguda FirstName, LastName ve Title verilerini çekeceğim, ancak verileri çektiğim tabloda FirstName ve LastName kolonlarını tek kolonda listeleyeceğim.

|  |  |
| --- | --- |
| 1  2 | SELECT FirstName + ' ' + LastName AS FirstAndLastName, Title  FROM Employees |

Burada ki deyim de, SELECT ile FirstName ve LastName i çekiyorum fakat birbirine + operatörü ile bağlayıp AS deyimi ile birlikte FirstAndLastName adlı bir kolon olarak listelenmesini sağlıyorum. Deyim sonucunda karşıma çıkan tablo:

[](http://www.ismailgursoy.com.tr/wp-content/uploads/2012/02/29.025.png)

Böylelikle normalde olmayan bir kolonu + operatörü ile oluşturmuş oluyorum.

**– Operatörü:** İki sayıyı birbirinden çıkarma işlemi için kullanılır.

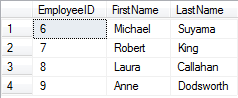
**\* Operatörü:** Bir sayıyı bir başka sayı ile çarpma için kullanılır.

**/ Operatörü:** Bir sayıyı bir diğer sayıya bölmek için kullanılır.

**> Operatörü: Büyüktür** operatörüdür. WHERE deyimi ile birlikte kullanılır, ilk değerin ikincisinden büyüklüğünü kontrol eder. Örneğin, Northwind veritabanında Employees tablosunda EmployeeID’si 5’ten büyük olanları listeleyelim:

|  |  |
| --- | --- |
| 1  2  3 | SELECT EmployeeID,FirstName, LastName  FROM Employees  WHERE EmployeeID > 5 |

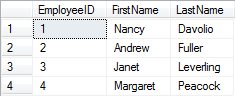
Bunun sonucunda alınan tablo:

[](http://www.ismailgursoy.com.tr/wp-content/uploads/2012/02/29.026.png)

**< Operatörü:** Küçüktür operatörüdür. WHERE deyimi ile birlikte kullanılır, ilk değerin ikincisinden küçüklüğünü kontrol eder. Örneğin, Northwind veritabanında Employees tablosunda EmployeeID’si 5’ten küçük olanları listeleyelim:

|  |  |
| --- | --- |
| 1  2  3 | SELECT EmployeeID,FirstName, LastName  FROM Employees  WHERE EmployeeID < 5 |

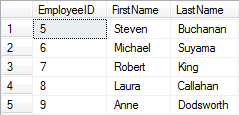
Bunun sonucunda alınan tablo:

[](http://www.ismailgursoy.com.tr/wp-content/uploads/2012/02/29.027.png)

**>= Operatörü: Büyük** eşittir operatörüdür. WHERE deyimi ile birlikte kullanılır, ilk değerin ikincisinden büyüklüğünü ve ona eşitliğini kontrol eder. Örneğin, Northwind veritabanında Employees tablosunda EmployeeID’si 5’e eşit ve 5’ten büyük olanları listeleyelim:

|  |  |
| --- | --- |
| 1  2  3 | SELECT EmployeeID,FirstName, LastName  FROM Employees  WHERE EmployeeID >= 5 |

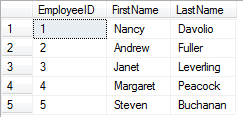
Bunun sonucunda alınan tablo:

[](http://www.ismailgursoy.com.tr/wp-content/uploads/2012/02/29.028.png)

**<= Operatörü: Küçük** eşittir operatörüdür. WHERE deyimi ile birlikte kullanılır, ilk değerin ikincisinden küçüklüğünü ve ona eşitliğini kontrol eder. Örneğin, Northwind veri tabanında Employees tablosunda EmployeeID’si 5’e eşit ve 5’ten küçük olanları listeleyelim:

|  |  |
| --- | --- |
| 1  2  3 | SELECT EmployeeID,FirstName, LastName  FROM Employees  WHERE EmployeeID <= 5 |

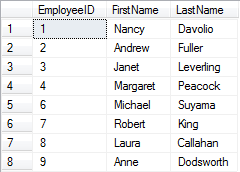
Bunun sonucunda alınan tablo:

[](http://www.ismailgursoy.com.tr/wp-content/uploads/2012/02/29.029.png)

**<>, != Operatörü:** İki türde de kullanılabilir, <>, != gibi. Eşit değildir operatörüdür. Belirtilen ilk değerin ikincisine eşit olmadığını kontrol eder. Örneğin yine Northwind veritabanında EmployeeID’si 5’e eşit olmayanları listeleyelim.

|  |  |
| --- | --- |
| 1  2 | SELECT EmployeeID,FirstName, LastName FROM Employees  WHERE EmployeeID != 5 |

Burada != yerine <> kullanırsak da aynı sonucu alırız ve aynı ifadeyi demiş oluruz. Bunun sonucunda alınan tablo:

[](http://www.ismailgursoy.com.tr/wp-content/uploads/2012/02/29.030.png)

**= Operatörü: Eşitlik** operatörüdür. Örneğin Northwind veri tabanında EmployeeID’si 5’e eşit olan veriyi listelemek için şöyle bir deyim yazabilirim.

|  |  |
| --- | --- |
| 1  2 | SELECT EmployeeID ,FirstName, LastName FROM Employees  WHERE EmployeeID = 5 |

**OR Operatörü:** SELECT deyimi ile birlikte kullanılır. Örneğin bir deyim ile sorgulama yaparken birden fazla durumu belirtebiliriz. Herhangi bir duyuma uyması yeterlidir. Örneğin şu deyim ile birlikte:

|  |  |
| --- | --- |
| 1  2 | SELECT EmployeeID,FirstName, LastName FROM Employees  WHERE EmployeeID = 5 OR EmployeeID = 3 |

Bu deyim ile birlikte karşıma EmployeeID’si 5 veya 3 olan veriler listelenir.

**AND Operatörü:** SELECT deyimi ile birlikte kullanılır. Belirtilen tüm durumların geçerli olmasını kontrol eder. Yani tüm şartlar sağlanmalıdır.

**NOT Operatörü:** LIKE deyiminin tersidir aslında. Bir durumum benzememe halini kontrol eder. Örneğin, Northwind veritabanında FirstName alanında ‘an’ geçmeyenleri listeleyelim.

|  |  |
| --- | --- |
| 1  2 | SELECT EmployeeID, FirstName, LastName FROM Employees  WHERE FirstName NOT LIKE '%an%' |

**? , \_ Operatörü :** ? veya \_ olarak kullanılabilir. Underscore karakter olarak bilinirler. Tek bir karakter yerine geçer. Örneğin bir tabloda bir ürün listesi olduğunu düşünelim. Seri noları olsun ilk karakter bir harf diğer karakterler rakam olsun. Rakamlar Seri noyu tanımlamış olsun, harf ise o ürünün bölgesel yerini belirtmiş olsun. Aynı seri nolu farklı bölgelerdeki ürünleri listelemek istediğimizde şöyle bir sorgu çekilebilir.

|  |  |
| --- | --- |
| 1  2 | SELECT UrunID, UrunSeriNo, UrunAdi FROM Urunler  WHERE UrunSeriNo LIKE '?12345' |

? yerine \_ işareti de kullanılabilir.

**%, \* Operatörü :** \* veya % olarak kullanılabilir. Birden fazla karakter yerine geçer. Örneğin bir ürünler tablosunda seri nosunda AB ifadesi için ürünleri listeleyen bir sorgu için:

|  |  |
| --- | --- |
| 1  2 | SELECT UrunID, UrunSeriNo, UrunAdi FROM Urunler  WHERE UrunSeriNo LIKE '%AB%' |

% işareti yerine \* işareti de kullanılabilir.

**CASE FONKSİYONU**

SQL Serverda CASE fonksiyonu sorgu sonucu dönen bir kolonun değerine göre farklı işlemler yapabilmemize olanak sağlar. İstediğimiz bir kolonun değerini belirli şartları kontrol ederek yeni bir değerle değiştirmek için kullanabiliriz. Örneğin tablomuzdaki cinsiyet kolonundan ‘E’ geldiğinde sorgu sonucunda Erkek ‘B’ geldiğinde ise Bayan yazmasını istiyorsak CASE fonksiyonunu kullanmak en mantıklı ve en kolay yoldur. SQL’de CASE fonksiyonu iki şekilde kullanılır. Bunlar SIMPLE CASE ve SEARCHED CASE olarak adlandırılır.

**Simple Case:** Basit CASE fonksiyonu genelde şu amaç için kullanılır. Mevcut bir veri satırında yer alan değerin daha sonra istediğimiz şekilde görüntülenmesini sağlayabiliriz. Kullanım olarak kalıbı:

**CASE testifadesi  
WHEN [karşılaştırılacak\_ifade\_1] THEN [Geri\_Dönüş\_Değeri\_1]  
WHEN [karşılaştırılacak\_ifade\_2] THEN [Geri\_Dönüş\_Değeri\_2]  
ELSE karşılaştırılacak son değer END AS [yeni\_kolonbaşlığı]**

Şimdi olayı örnekleyelim. Böylelikle daha net anlaşılacak. NORTHWND veritabanı üzerinde çalışıyorum.

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT EmployeeID, LastName, FirstName, Country,  CASE Country WHEN 'USA' THEN 'Amerika'  WHEN 'UK' THEN 'İngiltere'  ELSE 'Bilinmiyor' END AS ÜLKE  FROM Employees |
|  |  |

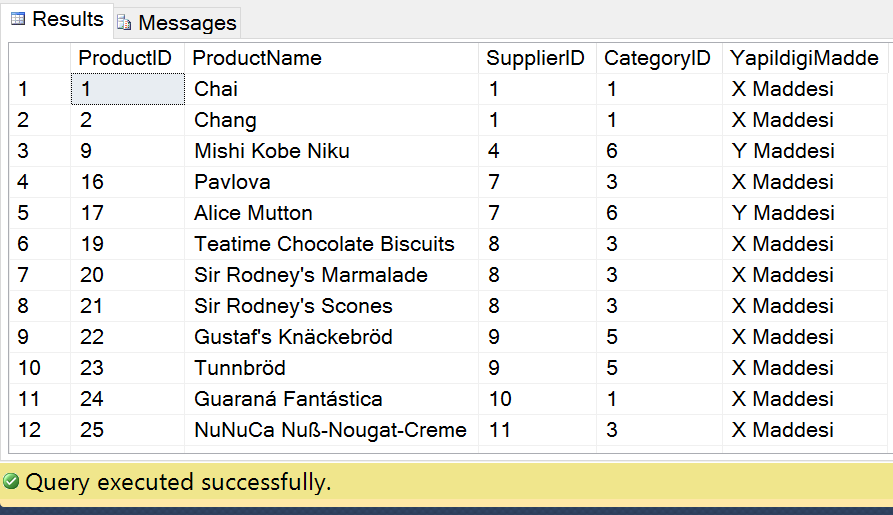
**Searched Case:** Searched Case eğer ifadeler oldukça karışık ise başvurulan bir yöntemdir. Kullanım kalıbı:

**CASE WHEN test\_edilecek\_ifade1 THEN değer1  
WHEN test\_edilecek\_ifade\_2 THEN değer2  
ELSE değer3 END**

Örnekleyelim.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT ProductID, ProductName, SupplierID, CategoryID,  CASE WHEN CategoryID IN (1,3,5) THEN 'X Maddesi'  WHEN CategoryID = 6 THEN 'Y Maddesi'  ELSE 'Bilinmiyor' END AS YapildigiMadde  FROM Products  WHERE CategoryID IN (1,3,5,6) |

Görüldüğü gibi bu sefer benzer şartlar aynı sonucu vereceği için bunları IN deyimi ile birlikte kullandık. Yani CASE fonksiyonunun searched edilmiş yani aranmış hali… Aynı şekilde burada da Products tablosunda CategoryID alan değeri 1,3,5,6 olanların CategoryID değeri 1,3,5 ise **X Maddesi** olarak eğer 6 ise **Y Maddesin** eğer hiç biri değilse **Bilinmiyor** bilgisini **YapildigiMadde** sütununa yazdıracaktır.



**CASE ile karşılaştırma yapmak:** CASE deyimi ile karşılaştırma da yapabiliriz. Örneğin şöyle bir tablonuz olduğunu düşünün. Tabloda seyaha bilgileri yer alıyor. İşte seyahat bilgilerinden kolonun birinde otelde kalma süresi bir diğeri yolda geçecek süre. Bu iki alandan hangisi büyük veya hangisi küçük bunu seçip ayrı bir kolonda tıpkı yeni bir tablo gibi ya da yeni oluşturulmuş bir alan gibi gösterebiliriz.

|  |  |
| --- | --- |
| 1  2  3 | SELECT VacationHours, SickLeaveHours, CASE WHEN VacationHours > SickLeaveHours  THEN VacationHours ELSE SickLeaveHours END AS 'Büyük Değer'  FROM HumanResources.Employee |

